Introduction to the GENI Cinema Architecture

GENI Cinema (GC) is a live video delivery service that operates using SDN and is realized using GENI compute and network resources on the GENI nationwide testbed. Being a geographically-distributed testbed, the GENI infrastructure lends itself nicely to GC in order to implement a content delivery network for the efficient delivery of video content to the customer at the edges. Combined with SDN, this allows both network and compute resources to be conserved while customers choose between the available video "channels" hosted by GC.

GC also places a soft quality of service guarantee on the video being streamed to each customer. With GC, there is a minimal delay when changing channels as well as high-definition audio and video quality of the stream itself. The architecture provides a "soft" guarantee, since each customer must connect to GC using the public Internet, or if they are on a participating campus they may use their campus network. The underlying assumption is that there must be sufficient network resources within the path the video traverses between GC to the customer over the public Internet or the campus network. Both of these networks are beyond the control of GC and are assumed to be capable of supporting video streaming (~1Mbps / stream).

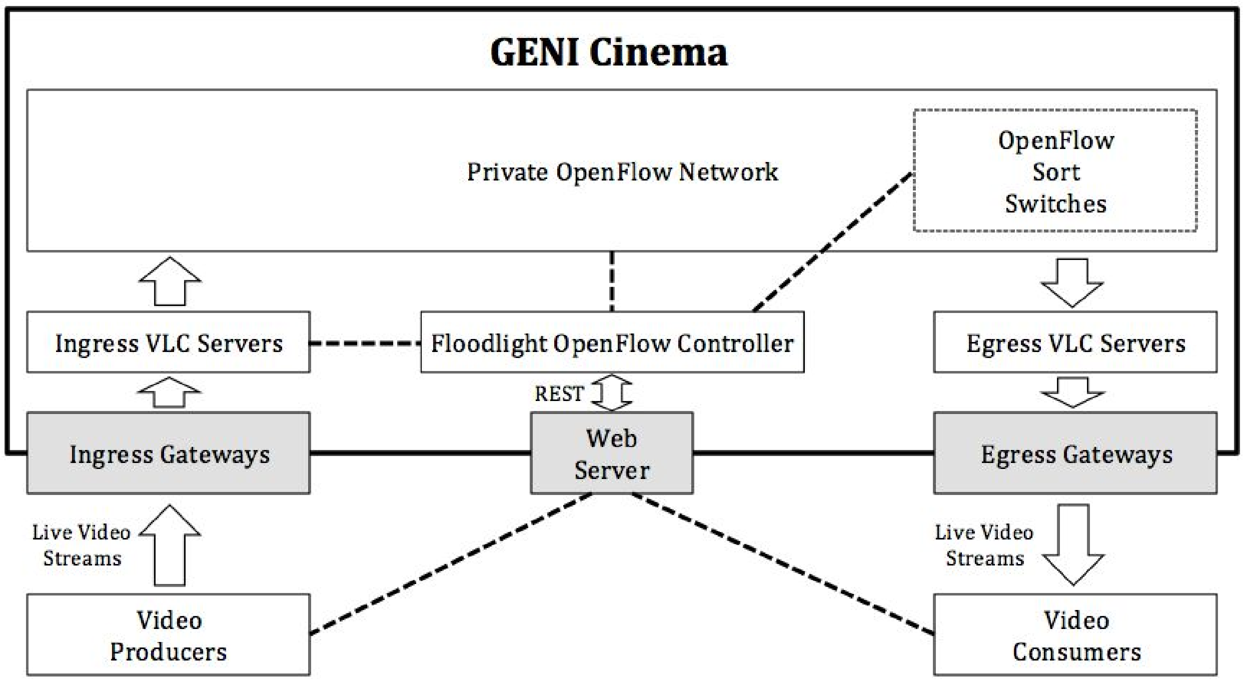


Figure 1: Logical components of the GENI Cinema architecture

As shown in **Figure 1**, GC consists of many functional blocks that come together to form the GC service. There are ingress and egress gateways for receiving and sending video streams, ingress and egress VideoLAN Client (VLC) servers for hosting video streams on the backend and providing them on the frontend, a Floodlight global resource and OpenFlow controller, both physical and software OpenFlow switches for controlling the flow of video streams internal to the GC service, and a public web server for customer interaction. The following sections describe these functional blocks and how they interact with each other.

The Web Server

The public web server hosts a website that allows video producers to specify live video streams as input to the GC service, as well as allows video consumers to subscribe to, watch, and switch between all available video content. The web server abstracts away all the low-level network and content management details from the customer to provide a straight-forward and seamless experience.

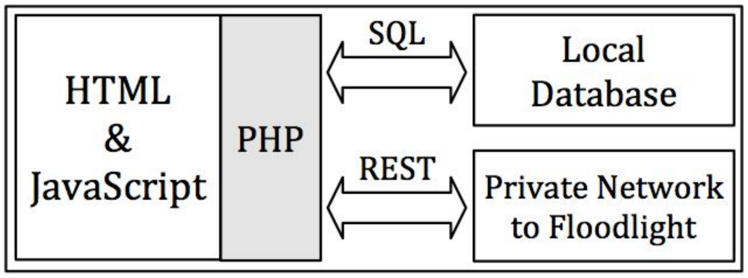


Figure 2: The public web server

The web server also detects video consumer disconnects, both graceful and ungraceful, and notifies the controller for resource reallocation. The web server is implemented with combination of HTML, PHP, and JavaScript, as depicted in **Figure 2**. A MySQL database on the backend serves as a repository of user account information and also caches current GC operating state information from the controller. To communicate with the Floodlight controller, the web server uses REST.

The OpenFlow Controller

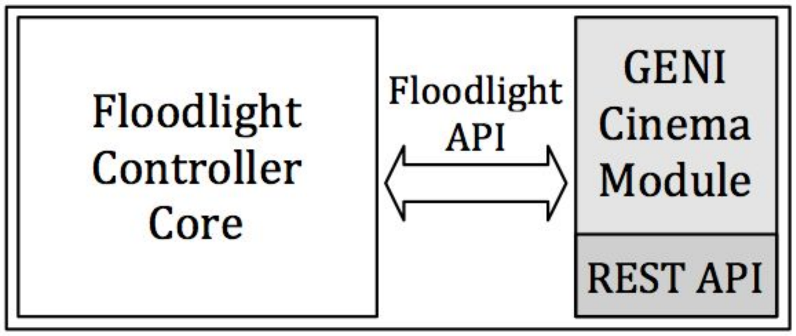


Figure 3: The Floodlight OpenFlow controller

The OpenFlow controller is implemented in Floodlight v1.1 (which supports OpenFlow 1.0 - 1.4). As shown in **Figure 3**, the Floodlight controller is equipped with a GC module responsible for managing all video content and routing it to the video consumers via the GC private network. The web server communicates to the Floodlight controller through its REST API in order to add, remove, modify, stream out (watch), and query for videos. All communication exchanged between the web server and Floodlight is done in JSON for portability and ease of use.

Floodlight’s REST API and the web server can also be used with HTTPS for added security in the event the JSON data must be relayed over an untrusted network or the controller’s REST API could be accessible by potentially untrusted clients.

Ingress and Egress Gateways

The points of video stream input into and output from the GC service are referred to as ingress and egress gateways. These gateways are implemented as nodes reserved at various aggregates in GENI and are designed to serve as firewalls and stream liveness detectors for video producers. These two functions have not been implemented but are included in the overall design of GC. The firewall serves to protect the internal GC components from unauthorized use and can be implemented using standard techniques like Linux IP tables and also with OpenFlow. The purpose of the stream liveness detectors is to determine if and when a video producer terminates a stream unexpectedly. In such an event, the GC service will react and free the resources for use by other video producers or take other appropriate actions. Liveness detectors can be implemented using OpenFlow 1.3 meters, OpenFlow 1.0 idle timeout notifications (send_flow_rem flag set in flow mods), or by periodic counter polling for applicable flows. These features were included in the design of GC, but have received a low priority and are not complete as of yet. As such, the *ingress and egress VLC servers* presently function as the ingress and egress gateways directly without the extra layer.

Ingress VLC Servers

When a video producer sends a video to GC, it must be hosted for video consumers to access. Furthermore, it must be accomplished such that the video producers and consumers do not directly interact with one another (i.e. for privacy and scalability reasons). As such, an ingress VLC server fulfills this requirement by receiving the video stream from the producer via the ingress gateway and hosting it for access by video consumer(s).

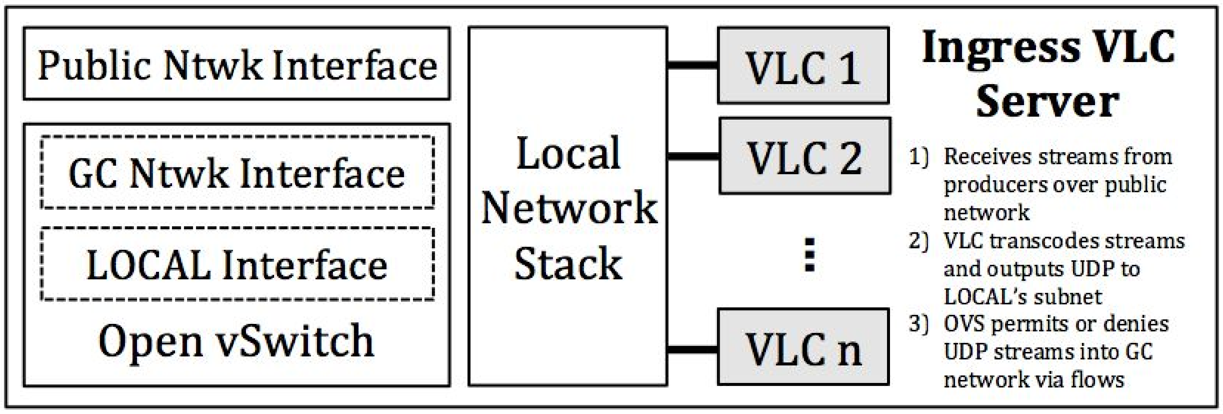

Public Ntwk Interface
GC Ntwk Interface
LOCAL Interface
Open vSwitch
Local Network Stack
VLC 1
VLC 2
VLC n
Ingress VLC Server
1) Receives streams from producers over public network
2) VLC transcodes streams and outputs UDP to LOCAL's subnet
3) OVS permits or denies UDP streams into GC network via flows

Figure 4: Software components of an ingress VLC server

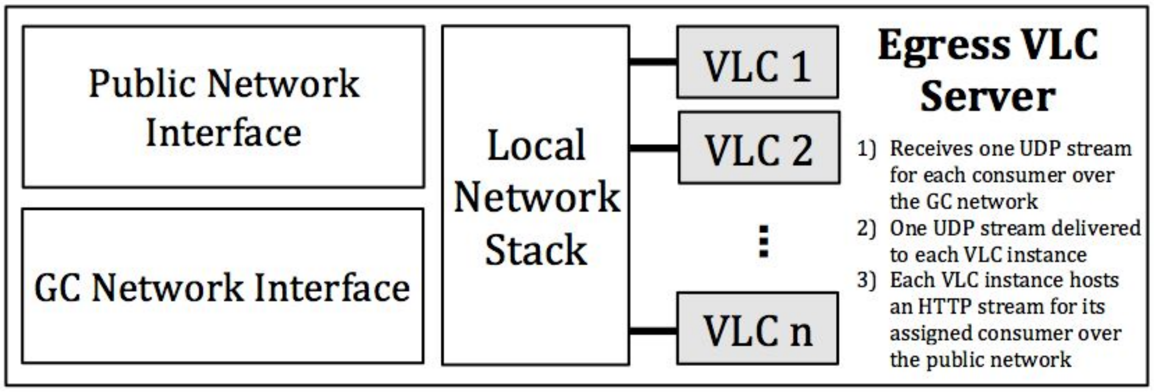
As shown in **Figure 4**, the ingress VLC server consists of multiple instances of VLC running in parallel. Each VLC instance listens on a transport port exposed to the video producers. As a result, each ingress VLC server can host as many live streams as there are VLC instances running.

After a VLC instance receives a video stream, it then transcodes the video stream to a constant format, and outputs the video stream into the private GC network over UDP. A constant format and encoding is used for each video stream in order to make the channel changing process more seamless to the video consumer in the same manner a television provider scales channels to the same aspect ratio and sometimes quality.

Also shown in **Figure 4**, each ingress VLC server also contains a single onboard Open vSwitch (OVS) instance. This OVS serves as an “on/off” switch for each video stream. It switches a channel “off” if there is presently no demand for the channel by any video consumers. Likewise, it switches a channel “on” if there is at least one consumer wishing to watch that particular channel. This is accomplished using the Floodlight controller and simple allow/drop OpenFlow flows in the OVS from it’s LOCAL port to the port attached to the private GC network -- one action per hosted video stream. The LOCAL port is the interface OVS uses to tap into the local network stack of the machine. The ingress VLC server routes packets from the VLC instances to this LOCAL port (exposed as an interface in the local network stack), which then allows the video packets to enter the private GC network (allow/drop flow permitting).

Egress VLC Servers

Like the ingress VLC servers, the egress VLC servers run copies of VLC in order to receive and send video streams. However, the egress VLC servers, unlike their ingress counterparts, interact directly with the video consumers instead of the video producers.



Public Network Interface
GC Network Interface
Local Network Stack
VLC 1
VLC 2
⋮
VLC n
Egress VLC Server
1) Receives one UDP stream for each consumer over the GC network
2) One UDP stream delivered to each VLC instance
3) Each VLC instance hosts an HTTP stream for its assigned consumer over the public network

Figure 5: Software components of an egress VLC server

Each egress VLC server consists of multiple instances of VLC running in parallel. Each VLC instance is assigned to a video consumer and listens on a transport port within the private GC network. The video stream the video consumer has chosen is directed to this transport port, is received by the VLC instance listening, and is served to the video consumer on the public-facing interface of the server. Each egress VLC server can host as many video consumers as there are VLC instances running.

GENI Cinema Private Network: Topology

The private OpenFlow network in GC is the link between the ingress and egress VLC servers. It is responsible for routing the video stream of each video hosted by an ingress VLC server to each video consumer wishing to watch that particular video on each egress VLC server. The topology of the private GC network is arbitrary; however, it must be capable of handling the bandwidth utilized by the video streams as they traverse the network from the ingress VLC servers to the egress VLC servers. In general, there are X ingress VLC servers and Y egress VLC servers. The network must be able to handle any arbitrary mapping (i.e. from 1-to-1 to 1-to-many) of video stream x in X to video consumer(s) y in Y. A fat tree topology with ingress and egress VLC servers at the edges naturally lends itself to this scenario, but any capable network will do ranging from a fat tree to a fully connected network.

It is important to note that a video stream will only traverse a link in the GC private network if there is some video consumer requesting that particular stream at the destination of that link. As such, guarantees can be placed on the upper and lower bandwidth consumption of each link, proving the scalability of the GC architecture.

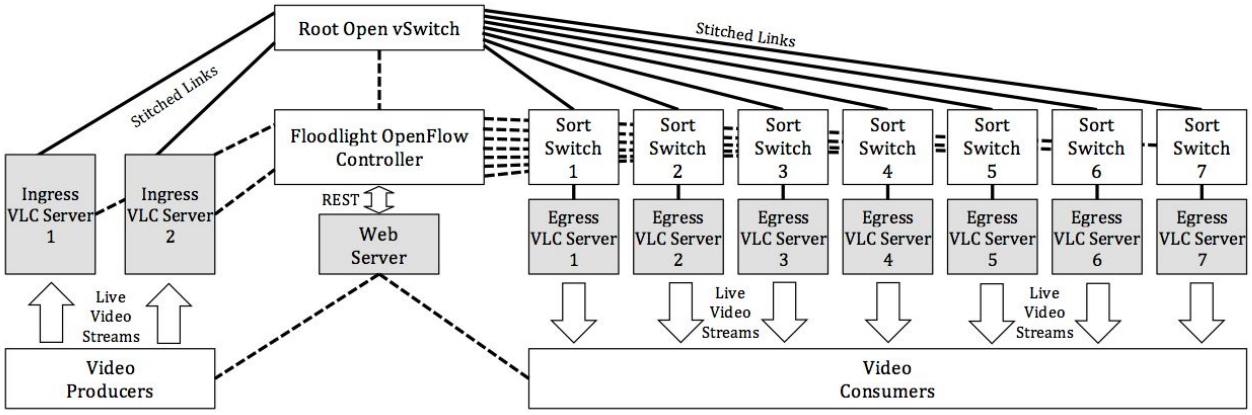


Figure 6: The GENI Cinema architecture and network topology for GEC22. Note that each stitched link also contains two physical OpenFlow switches under the control of Floodlight -- one at each end of the link.

As an example, at GEC22, a total of two ingress VLC servers and seven egress VLC servers were used (**Figure 6**). Each ingress and egress VLC server was hosted on dedicated InstaGENI raw PCs, where 100 VLC instances were run in parallel on each to support a total of 20 video producers and 700 video consumers. Each video stream was capped at 1Mbps such that an ingress VLC server could potentially receive a maximum of 10Mbps aggregate video traffic from video producers and thus send into the private GC network a maximum of 10Mbps. If more than 20 video producers are desired, additional ingress VLC servers can be added, each which would support an additional 10 video streams. (Alternatively, more VLC instances could be instantiated on each ingress VLC server.) Likewise, if more than 700 video consumers are needed, additional egress VLC servers can be added to GC, each which would support an additional 100 video streams.

Within the private, OpenFlow-based GC network, each video stream is intelligently routed from each ingress VLC server to every egress VLC server where there is presently demand for that particular stream. As such, a minimum of 0Mbps and a maximum of 10Mbps bandwidth is required leading from each ingress VLC server to the private GC network to satisfy the cases of zero demand and full-demand for the ingress VLC server's video streams, respectively. Similarly, a minimum of 1Mbps and a maximum of 20Mbps bandwidth is required to reach each egress VLC server for the cases where all video consumers are watching the same video stream and all are watching different streams, respectively. In this way, the GC architecture is scalable given that the network topology of the private GC network is designed to meet the guaranteed upper bound of each link.

GENI Cinema Private Network: Video Stream Routing, Duplicating, and Sorting

All video traffic output from the ingress VLC servers into the private GC network is unicast UDP in order to allow fast video stream switching without regard for the connection state, sequence, or source as TCP would impose. Each UDP video stream is directed through the network toward all egress VLC servers where there is at least one video consumer wishing to watch that particular stream. Prior to each egress VLC server is an OpenFlow switch called a "sort switch."

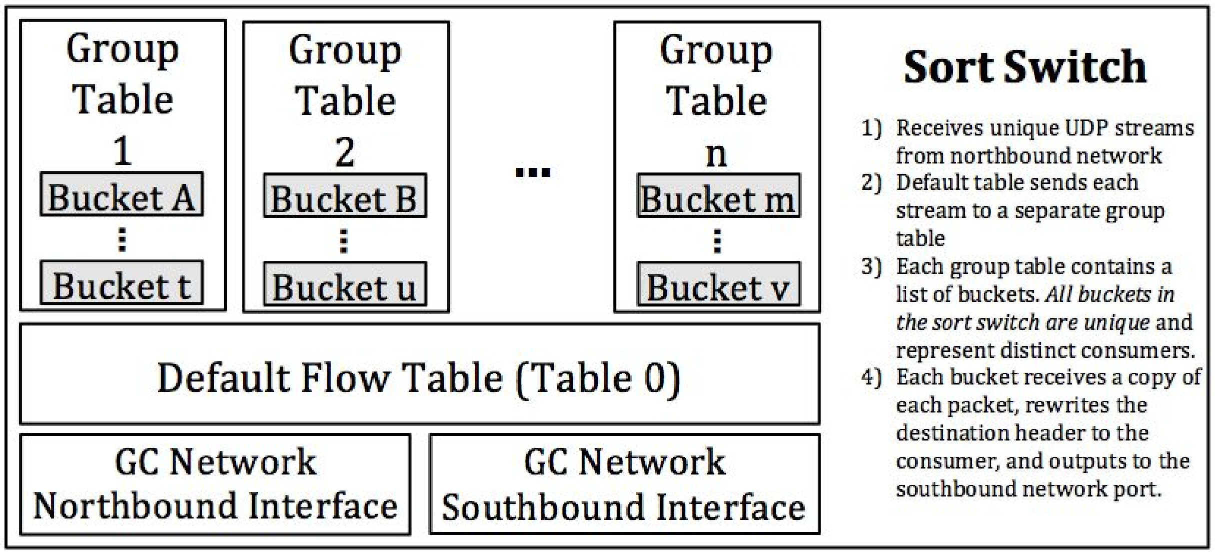

Group Table 1
Bucket A
Bucket t
Group Table 2
Bucket B
Bucket u
...
Group Table n
Bucket m
Bucket v
Default Flow Table (Table 0)
GC Network Northbound Interface
GC Network Southbound Interface
Sort Switch
1) Receives unique UDP streams from northbound network
2) Default table sends each stream to a separate group table
3) Each group table contains a list of buckets. All buckets in the sort switch are unique and represent distinct consumers.
4) Each bucket receives a copy of each packet, rewrites the destination header to the consumer, and outputs to the southbound network port.

Figure 7: The OpenFlow components of a GENI Cinema sort switch

This sort switch can be included on the VLC server itself as an OVS; however, it can also reside within the network. It is recommended that the sort switch be on a physically separate device from the egress VLC server, since both require real-time video stream processing. Each sort switch is responsible for taking the UDP video streams supplied as input on the northbound interface, duplicating these streams if appropriate, and sending them to the VLC instances on the associated egress VLC server. The involves rewriting the destination MAC, IP, and/or UDP port numbers in order for the network stack on the egress VLC server to accept the packets and send them to the VLC instances running as applications.

The sort switch helps to make the GC architecture scale more easily. It reduces the duplicate transmission of video streams until the last hop at the egress point where the consumers are connected. Continuing the GEC22 example, if there are 100 video consumers on a particular egress VLC server and all 100 video consumers wish to watch the same video stream, a single stream will be sent by the private GC network to the sort switch, using 1Mbps bandwidth. This single stream will be made into 100 copies where each copy's destination headers are rewritten such that the packets are routed to the VLC instance of each video consumer on the associated and nearby egress VLC server. This means 1Mbps of traffic enters the sort switch and 100Mbps exits. On the contrary, if there are 100 video consumers and each wishes to watch a different video of those available, there will be 20 unique video streams (the total channels available) entering the sort switch for a total of 20Mbps. The sort switch will make copies of each stream and rewrite the destination headers of each stream to send it to the VLC instance of the video consumer that wishes to watch that particular stream. After duplication, the total exit traffic is still 100Mbps leaving sort switch. The exit traffic is directly proportional to the number of video consumers presently attached to that particular egress gateway. The traffic entering can be no more than the minimum of either the total number of video channels available or the number of video consumers connected to the associated egress gateway.

As described, when a video consumer selects a channel to watch, the sort switch is responsible for selecting the appropriate input stream and sending it to the VLC instance on the egress VLC server associated with that video consumer. OpenFlow 1.1+ groups and buckets are used on the sort switch to implement this channel changing feature. As shown in **Figure 7**, every video is classified as an OpenFlow group, and every video consumer has a unique and single OpenFlow bucket. An OpenFlow bucket is a list of actions, and in the case of the buckets used for GC, each action list rewrites the destination MAC, IP, and/or UDP port headers of the packets. If a video consumer switches video channels, its bucket is removed from the previous video group of the channel it was viewing, and the bucket is simply added to the group of the video channel it wishes to watch. In this way, only one connection and video stream per customer is ever present at a given time within the private GC network, and zero connections are set up or torn down on the ingress and egress VLC servers upon a channel change. This optimizes the bandwidth usage within the private GC network, as well as reduces the load on the server resources during frequent channel changes.

Some Results

With a 1 second buffer at the user's video player (VLC web plugin), the average visual and auditory delay experienced when switching live, high-definition 720p streams is approximately 2 seconds. This delay is on par with and sometimes less than delays experimentally derived from modern digital cable and satellite television services (also ~2s). In addition, when switching between high-definition 720p streams, GC outperforms contemporary, non-prebuffered live video streaming solutions for the Internet such as YouTube Live (~5s).

Workflow

When a video producer wishes to register a video as input into GC, it does so via the website. The website relays the request to Floodlight, which responds with an appropriate ingress gateway IP address and transport port number to which the video producer can connect and send the video. Upon reception of the video stream, the ingress gateway relays it to an ingress VLC server where the live stream is hosted. (At present, all video streams are input to the GC service over UDP to provide the best quality of service. However, other protocols may be used with modification to the decoder at the host VLC server and the encoder at the video producer.)

When a video consumer on the website requests to watch a particular video, the website relays the request to Floodlight, which responds with an appropriate egress gateway IP address, and transport port number where the video consumer can connect and watch the video. The video selected is routed, duplicated, and rewritten within the private network of GC from the ingress VLC server on which it is being hosted to the private interface of the egress VLC server where the customer is connected. A VLC instance on this egress VLC server outputs the video on the public interface and relays it to the video consumer. (Video requests are presently served over HTTP in order to display the live stream within a VLC plugin in the video consumer's web browser. However, other protocols may be used to suit other video players at the video consumer. This would require modifying the egress gateway's VLC server parameters to use the appropriate encoding and transmission protocol as well.)